Appendix A

ADDLW ADD literal to W [label] ADDLW k Syntax: 0 ≤ k ≤ 255 Operands: Operation: $(W) + k \rightarrow W$ N,OV, C, DC, Z Status Affected: kkkk kkkk 0000 1111 Encoding: The contents of W are added to the 8-Description: bit literal 'k' and the result is placed in W. 1 Words: 1

Q Cycle Activity:

Cycles:

Q3 **Q4 Q2** Qi Write to W **Process** Decode Read Data literal 'k'

Example:

0x15 ADDLW

Before Instruction 0x10 W

After Instruction

0x25 W

ADD W to f **ADDWF** [label] ADDWF Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ Operation: (W) + (f) \rightarrow dest N,OV, C, DC, Z Status Affected: ffff 0010 01da Encoding: ' Add W to register 'f'. If 'd' is 0 the result Description: is stored in W. If 'd' is 1 the result is stored back in register "f (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default). 1 Words: 1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ADDWF

0,0 REG,

Before Instruction

0x17 W

0xC2 REG

After Instruction

0xD9 W REG 0xC2

ADD W and Carry bit to f **ADDWFC** [label] ADDWFC f,d,a Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(W) + (f) + (C) \rightarrow dest$ Operation: N,OV, C, DC, Z Status Affected: ffff ffff 00da 0010 **Encoding:** Add W, the Carry Flag and data memory Description: location 'T. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will

not be overridden.

Words: Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ADDWFC

REG, 0, 1

Before Instruction

Carry bit = 1 REG = 0x02 W = 0x4D

After Instruction

Carry bit = 0REG = 0x02W = 0x50 ANDLW AND literal with W
Syntax: [label] ANDLW

Operands: $0 \le k \le 255$

Operation: (W) AND. $k \rightarrow W$

1

Status Affected: N,Z

Encoding: , 0000 1011 kkkk kkkk

Description: The contents of W are AND'ed with the

Description: The contents of W are AND ed with the 8-bit literal 'k'. The result is placed in W.

Words:

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal		Write to W
<u> </u>	κ	Data	

Example:

ANDLW

0x5F

Before Instruction

W = 0xA3

After Instruction

W = 0x03

AND W with f ANDWF [label] ANDWF f,d,a Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ Operation: (W) .AND. (f) \rightarrow dest N,Z Status Affected: ffff ffff 0001 01da Encoding: The contents of W are AND'ed with reg-Description: ister 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default). Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ANDWF

REG, 0, 0

Before Instruction

W 0x17

0xC2 REG

After Instruction

0x02 W REG 0xC2

Branch if Carry BC

Syntax:

[label] BC n

Operands:

-128 ≤ n ≤ 127

Operation:

Encoding:

if carry bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected:

None

0010 nnnn 1110

Description:

If the Carry bit is '1', then the program

will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

	Q1	Q2	Q3	Q4
.	Decode	Read literal	Process Data	Write to PC
	No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

BC

Before Instruction

PC

address (HERE)

After Instruction

If Carry

address (HERE+12)

If Carry PC

address (HERE+2)

BCF	Bit Clear	f			
Syntax:	[label] B	CF f,t	o,a		
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$	5			
Operation:	0 → f 				
Status Affected:	None				
Encoding:	1001	bbba	ffff	ffff	
Description:	Bit 'b' in register 'f' is cleared. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1	•	-		
Q Cycle Activity:					
Q1	Q2	Q3_	Q	4	
Decode	Read	Proc	ess	Write	

register "

Example: BCF

7, 0 FLAG_REG,

register "f

Data

Before Instruction FLAG_REG = 0xC7 After Instruction FLAG_REG = 0x47

BN	Branch if Negative				
Syntax:	[label] BN n				
Operands:	-128 ≤ n ≤ 127				
Operation:	if negative bit is '1' (PC) + 2 + 2n \rightarrow PC				
Status Affected:	None				
Encoding:	1110 0110 nnnn nnnn				
Description:	If the Negative bit is '1', then the pro- gram will branch. The 2's complement number '2n' is				
	added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.				
Words:	1				
Cycles:	1(2)				

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	No operation

Example:

HERE

Jump

Before Instruction

PC

address (HERE)

After Instruction

If Negative PC If Negative PC

1; address (Jump)

0; address (HERE+2)

Branch if Not Carry BNC [label] BNC Syntax: $-128 \le n \le 127$ Operands: if carry bit is '0' Operation: $(PC) + 2 + 2n \rightarrow PC$ Status Affected: None nnnn mm 1110 0011 Encoding: If the Carry bit is '0', then the program Description: will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction. Words: 1(2) Cycles: Q Cycle Activity: If Jump: Q4 Q2 Q3 Q1 Write to PC **Process** Read literal Decode Data No

	No operation	No operation	No operation	operation	
If N	lo Jump:	-			
	Q1	Q2	Q3	Q4	7
	Decode	Read literal	Process	No	1

Decode	'n'	Data		operation
Example:	HERE	BNC	Jump	

Before Instruction
PC = address (HERE)

After Instruction
If Carry = 0:

If Carry = 0; PC = address (Jump)

Carry = 1; PC = address (HERE+2) BNN Branch if Not Negative

Syntax: [label] BNN. n: Operands: -128 ≤ n ≤ 127

Operands: $-128 \le n \le 127$ Operation: if negative bit is '0'

(PC) $+ 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 1110 0111 nnnn nnnn

Description: If the Negative bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be

PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1(2)

Q Cycle Activity: If Jump:

Cycles:

Q1 Q2 Q3 Q4

Decode Read literal Process Write to PC
'n' Data

No. No. No.

	, 'n'	Data	l
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
300000	'n'	Data	operation

Example: HERE BNN Jump

Before Instruction

PC = address (HERE)

After Instruction

If Negative = 0; PC = address (Jump)

If Negative = 1; PC = address (HERE+2)

			lat Osa		
INV		Branch if I		ertiow	
yntax	C	[label] BN	1V n		
)pera	nds:	-128 ≤ n ≤	127		
Opera	tion:	if overflow bit is '0' $(PC) + 2 + 2n \rightarrow PC$			
Status	Affected:	None			
Encod	iing:	1110	0101	nnnn	nnnn
Descr	iption:	If the Overfl gram will br		'0', the	n the pro-
	The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.			PC will the next s will be	
Word	ls:	1.			
Cycle	es:	1(2)			
Q Cy If Jui	cle Activity:				
Q1		Q2	Q3		Q4
:	Decode	Read literal	1	cess ata	Write to PC
	No	No		lo	No
	operation	operation	oper	ation	operation
lf N	lo Jump:	-			

Q3

BNV

address (HERE)

address (Jump)

address (HERE+2)

Process

Data

Jump

Q2

Read literal

'n

HERE

Q1

Example:

Decode

PC
After Instruction
If Overflow

Before Instruction

PC If Overflow PC Q4

No

operation

If N	o Jump:			
	Q1	Q2	Q3	Q4
	Decode	Read literal	Process Data	No operation
Exa	mple:	HERE	BNZ Jun	np
	Before Inst PC		nddress (HER	E)
	After Instru);·	

Branch if Not Zero

 $(PC) + 2 + 2n \rightarrow PC$

0001

If the Zero bit is '0', then the program

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

Q3

Process

Data No

operation

address (Jump)

address (HERE+2)

nnnn

nnnn

Q4

Write to PC

No

operation

[label] BNZ n

-128 ≤ n ≤ 127

if zero bit is '0'

None

1

Q2

Read literal

'n'

No

operation

1(2)

1110

will branch.

two-cycle instruction.

BNZ

Syntax:

Operands:

Operation:

Status Affected: Encoding:

Description:

Words:

Cycles:

Q Cycle Activity: If Jump:

Q1

Decode

No

operation

Unconditional Branch BRA [label] BRA n Syntax: -1024 ≤ n ≤ 1023 Operands: $(PC) + 2 + 2n \rightarrow PC$ Operation: None Status Affected: nnnn nnnn 1101 0nnn Encoding: Add the 2's complement number '2n' to Description: the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words:

Cycles:

2

Q Cycle Activity:

-	Q1 ·	Q2	Q3	Q4
	Decode	Read literal	Process Data	Write to PC
Ì	No operation	No operation	No operation	No operation

Example:

HERE

Jump BRA

Before Instruction

PC

address (HERE)

After Instruction

PC

address (Jump)

BSF	Bit Set f		
Syntax:	[label] BSF	f,b,a	
Operands:	0 ≤ f ≤ 255		
	0 <h<7< td=""><td></td></h<7<>		

 $a \in [0,1]$

1 → f Operation:

None Status Affected:

1000 **Encoding:**

ffff ffff bbba Bit 'b' in register 'f' is set. If 'a' is 0 Vir-Description: tual bank will be selected, overriding the

BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.

Words:

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Deco	de Read register	Process P Data	Write register "
1	10giolei		

Example:

FLAG_REG, 7, 1

Before Instruction

FLAG_REG=

0x0A

After Instruction

FLAG_REG=

A8x0

BTFSC	BTFSC Bit Test File, Skip if Clear			
Syntax:	[label] B	TFSC f,t	,a	
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
Operation:	skip if $(f < b >) = 0$			
Status Affected:	None			
Encoding:	1011	bbba	ffff	ffff
Description:	If bit 'b' in register 'f' is 0 then the next instruction is skipped. If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).			
Words: Cycles:	1 1(2) Note: 3 cycles if skip and followed by a 2-word instruction			

	-	., · · · ·		
Q Cycle Activity:				Q Cycle Ac
Q1	Q2	Q3	Q4	Q1
W 1	~~~			

	Decode	Read register 'f'	Process Data	No operation
lf ski	p:			

	Q1	Q2	Q3	Q4
	No operation	No operation	No operation	No operation
Į				

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

FLAG, 1, 0 HERE BTFSC

FALSE TRUE

Before Instruction

address (HERE) PC

After Instruction

If FLAG<1>

address (TRUE) AG<1> PC address (FALSE)

Bit Test File, Skip if Set **BTFSS**

[label] BTFSS f,b,a Syntax:

0 ≤ f ≤ 255 Operands:

0≤b<7 $a \in [0,1]$

skip if (f < b >) = 1Operation:

None Status Affected:

ffff ffff 1010 bbba Encoding:

If bit 'b' in register 'f' is 1 then the next Description: instruction is skipped.

If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a'

= 1, then the bank will be selected as per the BSR value (default).

1 Words:

1(2) Cycles:

Note: 3 cycles if skip and followed

by a 2-word instruction

ctivity:

Q1	Q2	Q3	Q4
Decode	Read	Process Data	No
	register 'f'		operation

If skip:

	Q1	Q2	Q3	Q4
- 1	No	No	No	No
į	operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

BTFSS

FLAG, 1, 0

FALSE TRUE

Before Instruction

address (HERE) PC

After Instruction

If FLAG<1>

address (FALSE)

AG<1>

1; address (TRUE)

Bit Toggle f BTG [label] BTG f,b,a Syntax: $0 \le f \le 255$ Operands: 0≤b<7 $a \in [0,1]$ ([45>) → f Operation: Status Affected: None ffff ffff bbba 0111 Encoding: Bit 'b' in data memory location 'f' is Description: inverted. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Decode Re regis	 Process Data	Write register 'f'

Example:

BTG PORTC, 4,

Before Instruction:

PORTC = 0111 0101 [0x75]

After Instruction:

PORTC = 0110 0101 [0x65]

BV	Branch if Overflow

Syntax: [label] BV

Operands: -128 ≤ n ≤ 127

Operation: if overflow bit is '1'

1

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 1110 0100 nmm

Description: If the Overflow bit is '1', then the pro-

gram will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a.

nnnn

two-cycle instruction.

Words:

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

BV Jump

Before Instruction

PC = address (HERE)

After Instruction

If Overflow = 1

PC = address (Jump)

If Overflow = 0

PC = address (HERE+2)

Branch if Zero BZ [label] BZ n Syntax: -128 ≤ n ≤ 127 Operands: if Zero bit is '1' Operation: $(PC) + 2 + 2n \rightarrow PC$ None Status Affected: nnnn 0000 nnnn 1110 **Encoding:** If the Zero bit is '1', then the program Description: will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

Jump

Before Instruction

PC

address (HERE)

After Instruction

If Zero

address (Jump) PC

If Zero

address (HERE+2) PC

CALL Subroutine Call

[label] CALL k,s Syntax:

0 ≤ k ≤ 1048575 Operands:

s ∈ [0,1]

 $(PC) + 4 \rightarrow TOS$, Operation:

 $k \rightarrow PC < 20:1>$

if s = 1

 $(W) \rightarrow WS$,

(STATUS) → STATUSS,

(BSR) → BSRS

None Status Affected:

Encoding: 1st word (k<7:0>)

2nd word(k<19:8>)

kkkk₀ k7kkk 1110 110s kkkka kkkk 1111 k₁₉kkk

Description:

Subroutine call of entire 2M byte memory range. First, return address (PC+4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-

cycle instruction.

Words:

2

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

THERE, Fast CALL

Before Instruction

PC = Address (HERE)

After Instruction

PC = TOS = Address (THERE)

Address (HERE + 4)

WS = BSRS= BSR

STATUSS = STATUS

Clear f CLRF [label] CLRF f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ $000h \rightarrow f$ Operation: $1 \rightarrow Z$ Status Affected: Z ffff ffff 101a 0110 Encoding: Clears the contents of the specified reg-Description: ister. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity: Q3 Q2 Q1 **Process** Write Read Decode register 'f'

CLRW	דם'	Clear Watchdog Timer				
Syntax	C	[label] C	LRWDT			
Opera	nds:	None				
Opera	tion:	000h → WDT, 000h → WDT postscaler, 1 → TO, 1 → PD				
Status	s Affected:	TO, PD				
Enco	ding:	0000	0000	0000	0100	
Desc	ription:	CLRWDT instruction resets the Watch- dog Timer. It also resets the postscale of the WDT. Status bits TO and PD ar set.				
Word	is:	1				
Cycle	es:	1 3				
QCy	cle Activity:					
	Q1	Q2	Q3	(Q4	
	Decode	No operation	Proc		No operation	

FLAG_REG, 1 CLRF Example: Before Instruction FLAG_REG 0x5A After Instruction 0x00 FLAG_REG

Data

register 4

Example: Before Instruction WDT counter After Instruction WDT counter WDT Postscaler TO PD 0x00 0

CLRWDT

COMF	Complem	ent f			
Syntax:	[label] C	OMF	f,d,a		
Operands:	$0 \le f \le 25$ $d \in [0,1]$ $a \in [0,1]$	5			
Operation:	$(\vec{f}) \rightarrow dc$	est			
Status Affected:	N,Z				
Encoding:	0001	11da	ffff	ffff	
Description:	The contents of register 'f are commented. If 'd' is 0 the result is stored but register 'f' (default). If 'a' is 0 Virtus bank will be selected, overriding the BSR value. If 'a' = 1, then the ban be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				

Q2	<u>u</u> 3	<u> </u>
Read register 'f'	Process Data	Write to destination
COME	REG. 0.	0
	UZ	register 'f' Data

Example: Before Instruction

Q Cycle Activity:

REG 0x13

After Instruction

0x13 REG 0xEC W

CPFSEQ	Compare f with W, skip if f = W			
Syntax:	[label] C	PFSEQ	f,a.	
Operands:	$0 \le f \le 255$ $a \in [0,1]$	5		
Operation:	(f) – (W), skip if (f) = (unsigned	•	on)	
Status Affected:	None			
Encoding:	0110	001a	ffff	ffff
Description:	Compares the contents of data memo location 'f' to the contents of W by per forming an unsigned subtraction. If 'f' = W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. 'a' = 1, then the bank will be selected per the BSR value (default).			
Words:	1			1

1(2) Cycles:

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

No	No	No
eration	operation_	operation
No	No operation	No operation
	no eration	110

Example:

CPFSEQ REG, 0 HERE

NEQUAL

EQUAL

Before Instruction

PC Address HERE ? W ? REG

After Instruction

W; If REG

Address (EQUAL) PC

W; If REG

Address (NEQUAL) PC

Compare f with W, skip if f > W **CPFSGT** [label] CPFSGT f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ (f) - (W),Operation: skip if (f) > (W)(unsigned comparison) None Status Affected: ffff 0110 010a ffff Encoding: Compares the contents of data memory Description: location 'T to the contents of the W by performing an unsigned subtraction. If the contents of 'f are greater than the contents of W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1(2) Cycles: Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register "	Data	operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

CPFSGT REG, 0

NGREATER

GREATER

Before Instruction

PC

Address (HERE)

W

?

After Instruction

If REG

W:

PC

Address (GREATER)

If REG

W:

PC

Address (NGREATER)

CPFSLT	Compare	f with W	, skip if	f < W
Syntax:	[label]	CPFSLT	f,a	
Operands:	$0 \le f \le 25$ $a \in [0,1]$	i 5		
Operation:	(f) - (W), skip if (f) < (W) (unsigned comparison)			
Status Affected:	None			
Encoding:	0110	000a	ffff	ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.			

If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation
	1 109.000		

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

CPFSLT REG, 1

NLESS

LESS

Before Instruction

Address (HERE) PC W

After Instruction

If REG

W:

PC

Address (LESS)

W: If REG ≥

PC

Address (NLESS)

DAW	Decima	ai A	djust W	Registe	r
Syntax: Operands: Operation:	(W< else (W< If [W<7:	0> > 3:0> <3:0:	-9] or [D(-) + 6 → ->) → W< ->9] or [C		1
Status Affected:	else (W≺ C	<7:4:	>) → W<	7:4>;	
Encoding:	000	0	0000	0000	0111
Description:	resultir variabl	ng fn les (d oduc	om the each in p	ght bit valu arlier addit acked BC rect packe	ion of two D format)
Words:	1				
Cycles:	1				
Q Cycle Activity:				_	

		res	ult.		
Word	is:	1			
Cycle	es:	1			
Q Cy	cle Activity:				
	Q1	Q2	_	Q3	Q4
	Decode		lead ster W	Process Data	Write W
Exa	mple1:	DA	M		
	Before Instr	uctio	n		
	W C DC	= = =	0xA5 0 0		

Example 2: Before Instruction

After Instruction

W = 0xCE C = 0 DC = 0

After Instruction

W = 0x34 C = 1 DC = 0

DECF	Decremen	t f		
Syntax:	[label] D	ECF f,	d,a	
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	•		
Operation:	$(f)-1\to c$	iest		
Status Affected:	C,DC,N,O	V,Z		
Encoding:	0000	01da	ffff	ffff
Description:	Decrement register T. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1.			
Q Cycle Activity:				
Q1	Q2	Q3	(<u> </u>
Decode	Read register 'f'	Proc		Write to destination
L				

Example:	DE	CF	CNT,	1, 0
Before Instru	ictio	n		
CNT Z	=	0x01 0		
After Instruc	tion			
CNT	=	0000		

DECFS		Decrement f			DCI
Syntax:		[label] DEC	CFSZ f,d,a		Syn
Operan	ds:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			Ope
Operati	on:	(f) $-1 \rightarrow des$ skip if result			Ор
Status	Affected:	None			Sta
Encodi	ng:		lda ffff	ffff	En
Descrip	otion:	ing it a two-c Virtual bank the BSR value	is 0 the result ne result is pla fault). : 0, the next in	is placed in ced back in struction, discarded, stead maken. If 'a' is 0 d, overriding nen the bank	De
Words	s:	1			W
Cycle		1(2) Note: 3 cyc by a 2	des if skip ar 2-word instru	nd followed	С
. O Cv	cle Activity:	•			C
_	Q1	Q2	Q3	Q4	
	Decode	Read register 'f'	Process Data	Write to destination	1
lf ski	o:	_		04	ľ
г	Q1	Q2	Q3	Q4 No	
	No operation	No operation	No operation	operation	
ا If ski		ved by 2-word			ļ
	Q1	Q2	Q3	Q4	
	No	No	No	No operation	
'	operation	operation No	operation	No	
	No operation	operation	operation	operation	
Exa	mple:	HERE	DECFSZ GOTO	CNT, 1, 1 LOOP	
		CONTINUI	Ε		
	Before Inst		ss (HERE)		
	After Instru CNT If CNT P If CNT	uction = CNT - = 0; C = Addre = 0;			

DCFSN	7	Decrement	f. skip il	not 0	1
Syntax:		label DCF			
Operand	ds:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation	on:	(f) $-1 \rightarrow de$ skip if result			
Status	Affected:	None			
Encodir	ng:	0100	llda :	ffff_	ffff
Descrip	otion:	The contents mented. If 'd' w. If 'd' is 1 register 'f' (d' if the result i	' is 0 the r the result efault).	esult is is plac	placed in ed back in
		tion, which is carded, and instead make tion. If 'a' is selected, ov 'a' = 1, then as per the E	s already an NOP it sing it a two O Virtual rerriding to the bank	fetched is exect to-cycle bank whe BSF will be	i, is dis- uted e instruc- vill be 3 value. If e selected
Words	: :	1			
Cycles		1(2) Note: 3 cy by a	cles if sk 2-word i	cip and instruc	i followed tion
-	cle Activity:	00	00		Q4
(Q1	Q2 Read	Q3 Proce		Write to
ļ	Decode	register "	Data	1	destination
lf skip);				
	Q1	Q2	Q3_		Q4
ſ	No	No	No		No operation
]	operation	operation ed by 2-wor	operat		operation
IT SKI	p and rollow Q1	Q2	Q3	,	Q4
i	No	No	No	,	No
	operation	operation	opera		operation
	No operation	No operation	opera		No operation
Exa	mple:	HERE ZERO NZERO	DCFSNZ :	; TEM	np, 1, 0
	Before Inst TEMP		= ?		
	After Instru TEMP If TEM P If TEM P	P C	= 0; = Add ≠ 0;	MP - 1, ress (? Iress ()	ZERO) NZERO)

Unconditional Branch **GOTO** [label] GOTO k Syntax: 0 ≤ k ≤ 1048575 Operands: k → PC<20:1> Operation: None Status Affected: Encoding: k7kkk $kkkk_0$ 1111 1st word (k<7:0>) 1110 kkkk k₁₉kkk kkkk₈ 1111 2nd word(k<19:8>)

Description:

Cycles:

GOTO allows an unconditional branch anywhere within entire 2M byte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-

cycle instruction.

Words: 2

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:

After Instruction

PC = Address (THERE)

GOTO THERE

Halt Processor HALT [label] HALT Syntax: None Operands: Processor halts execution after Operation: HALT instruction None Status Affected: 0001 0000 0000 0000 Encoding: While functioning in emulation mode, Description:

escription: While functioning in emulation mode, execution of the halt instruction will halt processor execution. Toggling the HALT pin or resetting (MCLR = 0) will

HALT pin or resetting (MCLR = 0) will bring the device out of halt, HALT instruction is not recognized in non-

emulation modes.

Words:

Cycles:

Q Cycle Activity:

Q1 Q2 Q3 Q4

	Q1	Q2	Q3	<u>Q4</u>	_
1	Decode	No	No	HALT	
į	-	operation	operation	<u> </u>	_

INCF	Increment f
Syntax:	[label] INCF f,d,a
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f) + 1 \rightarrow dest$
Status Affected:	C,DC,N,OV,Z
Encoding:	0010 10da ffff ffff
Description:	The contents of register 'f are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank who be selected as per the BSR value (default).
Words:	1
Cycles:	1

QCy	cle Activity:			
	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process Data	Write to destination

CNT, 1, 0

Example:	IN	ICF
Before Inst	ructio	n
CNT	=	0xFF
Z	=	0
Č	=	?
DC	=	?
After Instru	uction	
CNT	=	0x00

CNT	=	0x0
Z	=	1
C	=	1
DC	=	1

		•		
INCFSZ	Incremer	nt f, skip	if O	
Syntax:	[label]	INCFSZ	f,d,a	
Operands:	$0 \le f \le 25$ $d \in [0,1]$ $a \in [0,1]$	5		
Operation:	(f) + 1 \rightarrow skip if res			
Status Affected:	None			
Encoding:	0011	11da	ffff	ffff
Description:	The contents of register 'f are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f'. (default) If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value			

(defauit).

1 Words: 1(2) Cycles:

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decoc	-	Process	Write to
	register 'f'	Data	destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
ороламон			

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

	operation	operation	operation	<u>n</u>	operation
	No operation	No operation	No operatio	n	No operation
Exa	mple:	HERE NZERO	INCFSZ	<i>C7</i>	rr, 1, 0

Before Instruction Address (HERE)

PC After Instruction

CNT If CNT **CNT + 1**

ZERO

Address (ZERO)
0;
Address (NZERO)

Increment f, skip if not 0 INFSNZ INFSNZ f,d,a [label] Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ (f) + 1 \rightarrow dest, Operation: skip if result ≠ 0 None Status Affected:

Encoding:

0100	10da	ffff	ffff

Description:

The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).

If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value

(defauit).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination
l	register	1 0 4.4	

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

•	Q1	Q2	Q3	Q4
	No operation	No operation	No operation	No operation
	No operation	No operation	No operation	No operation

Example:

HERE ZERO

REG, 1. 0 INFSNZ

NZERO

Before Instruction

Address (HERE) PC

After Instruction

REG + 1 REG

If REG 0;

Address (NZERO)

Address (ZERO)

Inclusive OR literal with W IORLW

[label] IORLW k Syntax:

 $0 \le k \le 255$ Operands:

Operation: (W) OR. $k \rightarrow W$

N,Z Status Affected:

1001 kkkk kkkk 0000 Encoding:

The contents of W are OR'ed with the Description: eight bit literal 'k'. The result is placed in

W.

1 Words: 1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'K	Data	

Example:

IORLW

0x35

Before Instruction

W 0x9A

After Instruction

0xBF

Inclusive OR W with f IORWF IORWF f,d,a [label] Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ (W) .OR. (f) \rightarrow dest Operation: N,Z Status Affected: ffff ffff 0001 00da Encoding: Inclusive OR W with register 'f'. If 'd' is 0 Description: the result is placed in W. If 'd' is 1 the result is placed back in register T (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Write to destination

RESULT, 0, 1

Before Instruction

Example:

IORWF

RESULT = 0x13

0x91 W

After Instruction

RESULT = 0x13 0x93

W

Move f MOVF

[label] MOVF f,d,a Syntax:

 $0 \le f \le 255$ Operands:

 $d \in [0,1]$ $a \in [0,1]$

 $f \rightarrow dest$ Operation:

N,Z Status Affected:

0101 00da ffff Encoding:

ffff The contents of register 'f' is moved to a Description:

destination dependent upon the status of 'd'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value

(default).

Words:

1 1

Cycles:

Q Cycle Activity:

Q2	Q3	Q4
Read register "	Process Data	Write W
	Q2 Read register "f	Read Process

REG, 0, 0 MOVF Example:

Before Instruction

0x22 REG 0xFF

W

After Instruction

0x22 REG 0x22W

MOVFF

Move f to f

Syntax:

[label] MOVFF fs,fd

Operands:

 $0 \le f_s \le 4095$

 $0 \le f_d \le 4095$

Operation:

 $(f_s) \rightarrow f_d$ None

Status Affected:

Encoding: 1st word (source) 2nd word (destin.)

Description:

The contents of source register 'f_s' are moved to destination register 'fd'. Location of source 'fs' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'fd' can also be anywhere from 000h to

Either source or destination can be W (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVEF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words:

2

Cycles:

2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example:

MOVFF

REG1, REG2

Before Instruction

REG1 REG2

0x33 0x11

After Instruction

REG1 REG2

Move literal to low nibble in BSR MOVLB

Syntax:

[label]: MOVLB k.

Operands:

 $0 \le k \le 255$

Operation:

k → BSR

Status Affected:

None

Encoding:

0000 0001

Description:

The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words:

Cycles:

1 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write literal 'k' to BSR

Example:

MOVLB

Before Instruction

0x02 BSR register =

After Instruction

BSR register 0x05

Move literal to FSR LFSR [label] LFSR f,k Syntax: 0≤f≤2 Operands: 0 ≤ k ≤ 4095 $k \rightarrow FSRf$ Operation: Status Affected: None k₁₁kkk OOEE 1110 1110 Encoding: kkkk 0000 k7kkk 1111 The 12-bit literal 'k' is loaded into the Description: file select register pointed to by 'f' 2 Words: 2 Cycles: Q Cycle Activity:

'Q1	Q2	Q3	Q4
Decode	Read literal "k" MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal	Process Data	Write literal

Example:

LFSR 2, 0x3AB

After Instruction

FSR2H = 0x03 FSR2L = 0xAB MOVLW Move literal to W [label] MOVLW k Syntax: 0 ≤ k ≤ 255 Operands: Operation: $k \to W$ None Status Affected: kkkk 1110 kkkk 0000 Encoding: The eight bit literal 'k' is loaded into W. Description: Words: Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	<u></u>

Example:

MOVLW

0x5A

.

After Instruction

W = 0x5A

MOVWF Move W to f [label] MOVWF Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ Operation: $(W) \rightarrow f$ Status Affected: None ffff ffff 111a 0110 **Encoding:** Move data from W to register 'f'. Loca-Description: tion 'f can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words:

1

Cycles:

1

Q Cycle Activity:

	-	Q3	<u> </u>
Decode	Read	Process	Write
	egister 'f'	Data	register 'f'

Example:

MOVWF

REG, 0

Before Instruction

W = 0x4FREG = 0xFF

After Instruction

W = 0x4F REG = 0x4F

MULLW Multiply Literal with W

Syntax:

[label] MULLW

Operands:

 $0 \le k \le 255$

Operation:

(W) x k → PRODH:PRODL

Status Affected:

None

Encoding:
Description:

0000 1101 kkkk kkkk

An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

W is unchanged.

None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words:

1

Cycles:

Q Cycle Activity:

•	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example:

MULLW 0xC4

Before Instruction

W = 0xE2 PRODH = ? PRODL = ?

After Instruction

W = 0xE2 PRODH = 0xAD PRODL = 0x08

Multiply W with f MULWF [label] MULWF f;a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ (W) \times (f) \rightarrow PRODH:PRODL Operation: None Status Affected: ffff 001a ffff 0000 Encoding: An unsigned multiplication is carried Description: out between the contents of W and the register file location "f. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

REG, 1 Example: MULWF Before Instruction 0xC4 W REG PRODH 0xB5 PRODL After Instruction 0xC4 0xB5 REG 0x8A PRODH 0x94 PRODL

NEGF	Negate f	_		
Syntax:	[label] N	EGF	f,a	
Operands:	$0 \le f \le 25$ $a \in [0,1]$	5		
Operation:	(f)+1	•f		
Status Affected:	N,OV, C,	DC, Z		
Encoding:	0110	110a	ffff	ffff
Description:	plement. T memory lo bank will b	The result ocation 'f'. oe selecte e. If 'a' =	is placed If 'a' is 0 ed, overrio 1, then th	ding the ne bank will
Words:	1			•
Cycles:	. 1		•	
Q Cycle Activity:	•			
Q1	Q2	Q3		24
Decode	Read register 'f'	1	cess ata	Write register 'f'
<u> </u>	1 3			

REG, 1

0011 1010 [0x3A]

1100 0110 [0xC6]

NEGF

Before Instruction

REG

REG

After Instruction

Example:

No Operation NOP [label] NOP Syntax: None Operands: No operation Operation: None Status Affected: 0000 0000 0000 0000 Encoding: 3000K 200000 xxxx 1111 No operation. Description: 1 Words: Cycles: Q Cycle Activity: Q4 Q3 **Q2** Q1 No No No Decode operation operation operation

POP	Pop Top	of Retur	n Stack		
Syntax:	[label]	POP	-		
Operands:	None			-	
Operation:	(TOS) →	bit bucke	∍t		
Status Affected:	None				
Encoding:	0000	0000	0000	0110	
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.				
Words:	1				
Cycles:	1	•			
Q Cycle Activity:					
Q1	Q2	Q3	Q	4	

Q Cycle Acti

Example:

None.

operation value

No

POP

GOTO

Before Instruction

Decode

Example:

TOS = 0031A2h Stack (1 level down) = 014332h

POP TOS

NEW

No

operation

After Instruction

TOS = 014332h PC = NEW

Push Top of Return Stack **PUSH** [label] PUSH Syntax: None Operands: (PC+2) → TOS Operation: None Status Affected: 0101 0000 0000 0000 Encoding: The PC+2 is pushed onto the top of the Description: return stack. The previous TOS value is pushed down on the stack. This instruction allows to implement a software stack by modifying TOS, and then push it onto the return stack. Words:

QCy	cle Activity:		•	•
	Q1	Q2	Q3	Q4
	Decode	PUSH PC+2 onto return stack	No operation	No operation

Example:

Cycles:

PUSH

Before Instruction

TOS = 00345Ah PC = 000124h

After Instruction

PC = 000126h TOS = 000126h Stack (1 level down) = 00345Ah

RCALL	Branch S	ubroutii	18		
Syntax:	[label] RCALL n				
Operands:	-1024 ≤ n ≤ 1023				
Operation:	$(PC) + 2 \rightarrow TOS,$ $(PC) + 2 + 2n \rightarrow PC$				
Status Affected:	None				
Encoding:	1101	1nm	nnnn	nnnn	
	Subroutine call with a jump upto 1K				

Description:

Subroutine call with a jump upto 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-

cycle instruction.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address(Jump) TOS = Address (HERE+2)

RESET Reset [label] RESET Syntax: None Operands: Reset all registers and flags that Operation: are affected by a MCLR reset. All Status Affected: 0000 1111 1111 0000 Encoding: This instruction provides a way to exe-Description: cute a MCLR reset in software. 1 Words: Cycles:

Q3

No

operation

Q4

No

operation

Example:

Q Cycle Activity:

Decode

Q1

RESET

Start

reset

Q2

After Instruction

Registers = Reset Value Flags* = Reset Value

Return from Interrupt RETFIE RETFIE s [label] Syntax: $s \in [0,1]$ Operands: (TOS) \rightarrow PC, Operation: 1 → GIE/GIEH or PEIE/GIEL, if s = 1 $(WS) \rightarrow W$ $(STATUSS) \rightarrow STATUS,$ (BSRS) → BSR, PCLATU, PCLATH are unchanged. GIE/GIEH,PEIE/GIEL,STATUS reg. Status Affected: 0001 0000 0000 **Encoding:** Return from Interrupt. Stack is popped Description: and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If s' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, STA-TUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example:

RETFIE Fast

After Interrupt

PC = TOS W = WS BSR = BSRS STATUS = STATUSS GIE/GIEH, PEIE/GIEL = 1

Rotate Left f through Carry RLCF [label] RLCF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f < n >) \rightarrow dest < n+1 >$, Operation: $(f<7>) \rightarrow C$, (C) \rightarrow dest<0> C,N,Z Status Affected: 01da ffff ffff 0011 Encoding: The contents of register 'f' are rotated Description: one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). register f C 1 Words: Cycles: Q Cycle Activity: **Q4** Q3 **Q2** Q1

Read

register 'f

RLCF

1110 0110

1110 0110

1100 1100

Decode

Before Instruction

After Instruction

REG

W С

REG C

Example:

Write to

destination

Example:

Before Instruction

After Instruction

REG

REG

Process

Data

REG, 0, 0

RLNCF		Rotate Left f (no carry)			
Syntax:		[label]	RLNCF	f,d,a	3
Operand	ds:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$			
Operati	on:	, ,	dest<0>	1>,	
Status	Affected:	N,Z			
Encodi	ng:	0100	01da	fff	
Descrip		The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).			
Words	s:	1			
Cycle	s:	1			
Q Cyc	cle Activity:				
	Q1	Q2	Q3		Q4
	Decode	Read register 'f	Proc Da		Write to destination

RLNCF

1010 1011

0101 0111

REG, 1, 0

Rotate Right f through Carry RRCF [label] RRCF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<n>) \rightarrow dest<n-1>,$ Operation: $(f<0>)\rightarrow C,$ (C) → dest<7> C,N,Z Status Affected: ffff ffff 00da 0011 **Encoding:** The contents of register 'f' are rotated Description: one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). register f 1 Words: Cycles: Q Cycle Activity:

	1
Process Data	Write to destination

Example:

RRCF

REG, 0, 0

Before Instruction

REG = 1110 0110

C =

After Instruction

REG = 1110 0110 W = 0111 0011

C = 0

RRNCF	Rotate Right f (no carry))

Syntax: [label] RRNCF f,d,a

Operands: $0 \le f \le 255$

d ∈ [0,1]

a ∈ [0,1]

Operation: $(f < n >) \rightarrow dest < n-1 >$,

(f<0>) → dest<7>

Status Affected:

allo Allected.

d: <u>N,Z</u>

Encoding: 0100 00da fffff ffff

Description: The contents of register T are rotated

one bit to the right. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the

bank will be selected as per the BSR value (default).

register f

Words:

1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:

RRNCF REG, 1, 0

Before Instruction

REG = 1101 0111

After Instruction

REG = 1110 1011

Example 2:

RRNCF REG. 0,

Before Instruction

W =

REG = 1101 0111

After Instruction

W = 1110 1011 REG = 1101 0111

Set f SETF [label] SETF Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ $FFh \rightarrow f$ Operation: None Status Affected: ffff ffff 0110 100a Encoding: The contents of the specified register Description: are set to FFh. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity: Q3 Q4 **Q2** Q1 Write **Process** Decode Read register 'f' Data register " REG,1 SETF Example:

0x5A

OxFF

Before Instruction

After Instruction

REG

REG

SLEEP	Enter SLEEP mode				
Syntax:	[label] SLEEP				
Operands:	None				
Operation:	00h → WDT, 0 → WDT postscaler, 1 → TO, 0 → PD				
Status Affected:	TO, PD				
Encoding:	0000	0000	0000	0011	
Description:	The power-down status bit (PD) is cleared. The time-out status bit (TO) is set. Watchdog Timer and its postscaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.				
Words:	· 1	•			
Cycles:	1				
Q Cycle Activity:					
Q1	Q2	Q3	(Q4	
Decode	No operation	Proc Da	1	Go to sleep	

Example: SLEEP

Before Instruction

TO = ?
PD = ?

After Instruction

TO = 11 PD = 0

† If WDT causes wake-up, this bit is cleared

Subtract f from W with borrow **SUBFWB** [label] SUBFWB f,d,a. Syntax: 0≤f≤255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(W) - (f) - (\overline{C}) \rightarrow dest$ Operation: N,OV, C, DC, Z Status Affected: ffff 0101 ffff 01da Encoding: Subtract register 'P and carry flag (bor-Description: row) from W (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored in register 'f' (default) . If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (defauit). 1 Words:

1

Q2

Read

register 4

Q4

Write to

destination

Q3

Process

Data

Cycles:

Q Cycle Activity:

Decode

Q1

SUBFWB			
Example 1:	ຮບ	BFWB	REG, 1, 0
Before Instru	ıction	-	
REG	=	3	
W	=	2.	
С	=	1	
After Instruc			
REG	=	FF	
W	=	2	
ç	=	0	
C Z N	=	1	; result is negative
Example 2:	s	UBFWE	REG, 0, 0
Before Insti	ructio	า	•
REG	=	2	•
W	. =	5	
С	· =	1	·
After Instru	ction		•
REG	=	2	
W	=	3	
ç	=	1	
C Z N	=	ŏ	; result is positive
Example 3:		SUBFW	B REG, 1, 0
Before Ins	tructio	on ·	
REG	=	1	
W	=	2	•
С	=	0	•
After Instr	uction	1	
REG	=	0	
W	=		
ç	=		; result is zero
C Z N	=		, 1000it in and

Subtract W from literal **SUBLW** [label] SUBLW k Syntax: $0 \le k \le 255$ Operands: $k-(W) \rightarrow W$ Operation: N,OV, C, DC, Z Status Affected: kkkk kkkk 0000 1000 **Encoding:** W is subtracted from the eight bit lit-Description: eral 'k'. The result is placed in W. Words: Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1

Write to W **Process** Read Decode literal 'k' Data

0x02

Before Instruction

Example 1:

W С

After Instruction

; result is positive 0

SUBLW

CZN

SUBLW 0x02Example 2:

Before Instruction

2 W C

After Instruction

W

; result is zero CNN

0

Example 3:

SUBLW

Before Instruction

W C

After Instruction

; (2's complement) W ; result is negative

CZN

Subtract W from f SUBWF

[label] SUBWF f,d,a Syntax:

0≤f≤255 Operands:

 $d \in [0,1]$ $a \in [0,1]$

 $(f) - (W) \rightarrow dest$ Operation: N,OV, C, DC, Z

1

Status Affected: 0101

ffff ffff Encoding: Subtract W from register 'f' (2's com-Description: plement method). If 'd' is 0 the result is

stored in W. If 'd' is 1 the result is stored back in register "f" (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per

the BSR value (default).

Words:

1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register "	Process Data	Write to destination

Example 1:	SUBWF REG, 1, 0
Before Instru REG	ection = 3
W	= 2
С	= ?
After instruct REG	tion
W	= 2 = 1 ; result is positive
C Z N	= 0
	= 0 SUBWF REG, 0, 0
Example 2:	
Before Instru REG	= 2
w	= 2 = ?
C After Instruc	_ ·
REG	_
w	= 0 = 1 ; result is zero
C Z N	= 1 = 0
Example 3:	SUBWF REG. 1, 0
Before Inst	
REG	
W	= 2 = ?
After Instru	
REG W	= FFh ;(2's complement) = 2
	= 0 ; result is negative
C Z N	= 0 = 1
SUBWFB	Subtract W from f with Borrow
Syntax:	[label] SUBWFB f,d,a
Operands:	0 ≤ f ≤ 255
	$d \in [0,1]$ $a \in [0,1]$
Operation:	$(f) - (W) - (\overline{C}) \rightarrow \text{dest}$
Status Affecte	
Encoding:	0101 10da ffff ffff
Description:	Subtract W and the carry flag (borrow)
	from register 'f' (2's complement method). If 'd' is 0 the result is stored
	in W. If 'd' is 1 the result is stored back
	in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the
	BSR value. If 'a' is 1, then the bank
	will be selected as per the BSR value (default).
Words:	1
Cycles:	1
Cycles.	·

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 4	Data	destination

L			
SUBWFB			
Example 1:	SU	BWFB	REG, 1, 0
Before Instru	uction		
REG	=	0x19	(0001 1001)
W	=	0x0D	(0000 1101)
С	=	1	
After Instruc	tion		
REG	=	0x0C	
W	=	OxOD	(0000 1101)
Ç	=	1	
C Z N	=	Õ	; result is positive
Example2:	S	UBWFB	REG, 0, 0
Before Instr	uction	1	
REG		0x1B	(0001 1011)
w	=		(0001 1010)
Ċ	=	0	,
After Instru	ction		
REG	=	0x1B	(0001 1011)
W		0x00	
С	=	1	11.1.
C Z N	=	1	; result is zero
Example3:		•	REG, 1, 0
Before Inst	ructio	n	
REG	=	0x03	(0000 0011)
W	=	0x0E	(0000 1101)
, C	=	1	
After Instr	uction		-
REG	=		
W	=		(0000 1101)
C Z	=	0	
Z N	=	1	: result is negative

SWAPF Swap f [label] SWAPF f,d,a Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<3:0>) \rightarrow dest<7:4>$ Operation: $(f<7:4>) \rightarrow dest<3:0>$ None Status Affected: ffff ffff 10da 0011 Encoding: The upper and lower nibbles of register Description: 'f are exchanged. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	<u> </u>
Decode	Read register 'f'	Process Data	Write to destination

Example:

SWAPF REG, 1, 0

Before Instruction

REG = 0x53

After Instruction

REG = 0x35

Table Read TBLRD [label] TBLRD (*; *+; *-; +*) Syntax: None Operands: if TBLRD *, Operation: (Prog Mem (TBLPTR)) → TABLAT; TBLPTR - No Change; if TBLRD *+, (Prog Mem (TBLPTR)) \rightarrow TABLAT; (TBLPTR) +1 → TBLPTR; if TBLRD *-, $(\mathsf{Prog}\;\mathsf{Mem}\;(\mathsf{TBLPTR}))\to\mathsf{TABLAT};$ (TBLPTR) -1 → TBLPTR; if TBLRD +*, (TBLPTR) +1 \rightarrow TBLPTR; (Prog Mem (TBLPTR)) → TABLAT; Status Affected: None

Encoding:

0000	0000	0000	10nn
1	1		nn=0 *
1	1		=1 *+
1			=2 *-
			=3 +*
1			

Description:

There are four options with a TBLRD instruction to determine what happens to the 21-bit Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately, and the contents of the program memory location pointed to by the TBLPTR are loaded into the 8-bit Table Latch (TABLAT). The LSb of the TBLPTR selects which byte of the program memory location will be read. If LSb = 1, the high byte will be loaded into the TABLAT. If LSb = 0, the low byte will be loaded into the TABLAT.

1 Words: 2

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (OE goes low) TABLAT updated

	TBLRD	Table Read			_
•	Example1:	TBLRD *+	t.		•
	Before Instruc TABLAT TBLPTR MEMORY	tion (0x00A356)	=======================================	0x55 0x00A356 0x34	
	After Instruction TABLAT TBLPTR	on	==	0x34 0x00A357	
	Example2:	TBLRD +*	;		
;	Before Instru TABLAT TBLPTR MEMORY MEMORY	ction ((0x01A357) ((0x01A358)	= =	0xAA 0x01A357 0x12 0x34	-
:	After Instruct	ion		0.24	

TBLPTR

0x01A358

Table Write TBLWT TBLWT (*; *+; *-; +*) [label] Syntax: None Operands: if TBLWT *. Operation: (TABLAT) → Prog Mem(TBLPTR); TBLPTR - No Change; if TBLWT *+, $(TABLAT) \rightarrow Prog Mem(TBLPTR);$ (TBLPTR) +1 → TBLPTR; if TBLWT *-, (TABLAT) → Prog Mem(TBLPTR); (TBLPTR) -1 → TBLPTR; if TBLWT +*, (TBLPTR) +1 → TBLPTR; (TABLAT) → Prog Mem(TBLPTR);

Status Affected:

Encoding:

1	None	_		
[0000	0000	0000	11nn
١				nn=0 *
		ł	į	=1 *+
1			į	=2 *-
		1		=3 +*

Description:

There are four options with a TBLWT instruction. These options determine what happens to the Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately.

The contents of Table Latch (TABLAT) are written to the program memory location pointed to by TBLPTR.

If TBLPTR points to an external program memory location, then the instruction executes in two cycles.

Since the TABLAT is only one byte wide, a multiple of two TBLWT instructions must be executed to program internal memory locations. For example, if the device is defined to program one word at time, an internal memory location is programed in the following manner:

- 1) Set TBLPTR to an even byte
- 2) Write low byte to TABLAT
- 3) Execute TBLWT *+ (2-cycle)
- 4) Write high byte to TABLAT
- 5) Execute TBLWT *+ (long write)

A long write to an internal EPROM location is terminated when an interrupt is received. The post increment TBLWT instruction is the only TBLWT instruction that is recommended for writes to internal memory. (Writes to internal EPROM are only available on devices with 64 or more pins.)

TBLWT Table Write

Cycles: 2 (many if long write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)

Example1:

TBLWT *+:

Before Instruction

TABLAT = 0x55 TBLPTR = 0x00A356 MEMORY(0x00A356) = 0xFF

After Instructions (table write completion)

TABLAT = 0x55 TBLPTR = 0x00A357 MEMORY(0x00A356) = 0x55

Example 2:

TBLWT +*:

Before Instruction

TABLAT = 0x34
TBLPTR = 0x01389A
MEMORY(0x01389A) = 0xFF
MEMORY(0x01389B) = 0xFF

After Instruction (table write completion)

TABLAT = 0x34 TBLPTR = 0x01389B MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0x34

Debugger Subroutine Cail RAP [label] TRAP syntax: None Operands: $(PC) + 2 \rightarrow TOS,$ Operation: 000028h → PC<20:1>

INBUG Status Affected:

Encoding:

1110 0000 0000 0000

Debugger Trap to 00028h. First, return Description: address (PC+2) is pushed onto the return stack. Then the 20-bit value '000028h' is loaded into PC<20:1>. The INBUG status bit is set. TRAP is a

two-cycle instruction.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Push PC to stack	No operation	Write 000028h to PC
No operation	No operation	No operation	No operation

Example:

HERE

TRAP

Before Instruction

Address (HERE) PC =

After Instruction

PC = 00 TOS = A INBUG = 1 000028h

Address (HERE + 2)

TRET	Trap Ret	urn from	Subrou	line
Syntax:	[label]	TRET		
Operands:	None			
Operation:	(TOS) → PCLATU	PC I, PCLATI	d are und	hanged
Status Affected:	INBUG			
Encoding:	0000	0000	1110	0001
Description:	popped a is loaded	om debugg and the top into the pr	of the sta	ck (TOS)

Words:

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack
No operation	No operation	No operation	No operation

Example:

TRET

After Interrupt PC = TOS INBUG = 0

Test f, skip if 0 **TSTFSZ** [label] TSTFSZ f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ skip if f = 0Operation: Status Affected: None ffff ffff 0110 011a Encoding: If T = 0, the next instruction, fetched Description: during the current instruction execution, is discarded and a NOP is executed making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: 1(2) Cycles: Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

	Q1	Q2	Q3	Q4
	No operation	No operation	No operation	No operation
- 1				

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE TSTFSZ CNT, 1

NZERO

ZERO

Before Instruction

PC = Address(HERE)

After Instruction

If CNT = 0x00,

PC = Address (ZERO)

If $CNT \neq 0x00$.

PC = Address (NZERO)

XORLW	Exclusive OR literal with W				
Syntax:	[label] XORLW k.				
Operands:	0 ≤ k ≤ 255				
Operation:	(W) .XOR. $k \rightarrow W$				
Status Affected:	N,Z				
Encoding:	0000	1010	kkkk	kkkk	
Description:	The contents of W are XOR'ed with the 8-bit literal 'k'. The result is placed in W.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
01	02	O3-	Q	4	

Example:

XORLW 0xAF

Read

literal 'k'

Process

Data

Write to W

Before Instruction

Decode

W = 0xB5

After Instruction

W = 0x1A

y ;

-	DIME	
	HWF	

Exclusive OR W with f

Syntax:

[label] XORWF f,d,a

Operands:

0 ≤ f ≤ 255

 $d \in [0,1]$

 $a \in [0,1]$

Operation:

(W) .XOR. (f) \rightarrow dest

Status Affected:

N,Z

Encoding:

0001 10da ffff ffff

Description:

Exclusive OR the contents of W with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in the register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value

(default).

Words:

1

Cycles:

1

Q Cycle Activity:

Ω1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

XORWF REG, 1, 0

Before Instruction

REG = 0xAF

W = 0xB5

After Instruction

REG = 0x1A

W = 0xB5